CLEAN VERSION OF ALL REPLACEMENT PARAGRAPHS OF THE SPECIFICATION

Please replace the paragraph in the Summary of the Invention on page 3, lines 3-20, with the following amended paragraph:

--The present invention relates to a system and methodology of reducing process algebra (employed to facilitate modeling business workflow processes) to a language that facilitates modeling business workflow processes. A typical business workflow process in accordance with the present invention may include a plurality of business processes defined by a number of operations - the operations defining constraints on the business processes. The present invention provides for reducing any of a plurality of conventional process algebri to a model for business workflow applications. For example, combinators (a Π-calculus derivative) can be employed in modeling a business workflow application. The model is then reduced to an application programming language to allow users to choose between features of the present invention and conventional features associated with modeling application specific business processes. Preferably, the application program language is a scheduling language that may be represented as a graphical user interface program convertible to a schedule written in a programmable language. The present invention facilitates unsophisticated programmers in implementing modeling techniques for specific business workflow processes. The present invention further provides expression for viewing and grouping a workflow schedule separate from any binding associated with a specific implementation and a specific technology, which allows for a common business model to be utilized across a variety of implementations and technologies.--

Please replace the paragraph on page 12, beginning at line 23, and ending on page 13, line 12, with the following amended paragraph:

--In addition to the above stated features, the model allows for concurrent execution of actions within transactions. Transactions will not commit until a final action within a transaction has completed. The model also allows for explicit determination of transaction boundaries in addition to increased granularity of transactions. For example, transaction (T5) 50 has been defined as having four actions, while transaction (T3) 40 and (T4) 45 has been defined as including three and two actions, respectively. Transaction (T2) 30 has been defined as including transaction (T3) 40, (T4) 45 and (T5) 50, but can be defined as any two of transaction (T3) 40, (T4) 45 and (T5) 50 or simply any of transaction (T3) 40, (T4) 45 and (T5) 50. Therefore, the present invention allows for defining transaction boundaries and increasing granularity. Additionally, actions can happen concurrently independent of isolation because the data that the actions work on are independent of one another. Since isolation of the model has been relaxed to allow for increased granularity, transactions cannot simply be rolled back upon a failure of a single transaction, as is conventional. This is because the data associated with committed interdependent transactions is not locked after commitment, and therefore data may be compromised before another concurrent interdependent transaction fails. Therefore, the present invention allows for compensation to be invoked upon a failure of a transaction or an action. The compensation can be invoked to include compensating tasks for committed interdependent concurrent transactions and all committed transactions and actions outside the parent transaction. However, compensation upon a failure can include any compensating action to be invoked based on a particular application or desire.--

Please replace the paragraph on page 23, lines 3-17, with the following amended paragraph:

--A partition construct describes a collection of independent concurrent processes. The partition construct allows the users to represent transactions as autonomous independent transactions separate from concurrent interdependent transactions. In the present example, independent refers to the fact that each process in the partition refers to different ports, while concurrent meaning that all the processes in the partition proceed in parallel. Fig. 23a illustrates EBNF notation for a partition construct, while Fig. 23b illustrates the partition construct in XML. A graphical user interface component representing a partition is illustrated in Fig. 23c. A connect construct allows for modeling a simple form of communication between processes. Fig. 24a illustrates EBNF notation for a connect construct, while Fig. 24b illustrates the connect construct in XML. A graphical user interface component representing a connect construct is illustrated in Fig. 24c. The connect construct allows the users to connect processes. For example, if a source action having a port p and a message m occurs in a connected process that is connected to a sink action having a port q and a message n, the message m from the source action will be received by the sink action as a message n.--

Please replace the paragraph on page 27, lines 38-40 of the Appendix, with the following amended paragraph:

```
--< ! ELEMENT   context   (transactional?) >
< ![1] ATTLIST   context
   name        ID        #REQUIRED>--
```